Document Number	ENG-17247
Revision	D
Author	Fred Niemi
Project Manager	Tom George

EDS V1.0 & NESTOR V1.0

System Functional Specification

Project Headline

This specification covers both EDS and NESTOR. EDS is the Event Distribution System for CNMS. NESTOR is the event/exception monitor used by the CNMS.

Reviewers

Department	Name	Acceptance Date
IBU Network Management	Frank Pittenger (Software Engineer)	
IBU Network Management	Fred Niemi (Software Engineer)	
IBU Network Management	Tom George (Project Manager)	
IBU Network Management	Zbigniew Kielczewski(Software Engineer)	
Product Marketing - IBU	Lori Bush (Product Marketing)	
Network Management	John Ahlstrom (Software Engineer)	
Network Management	Stephen Schleimer (Software Engineer)	

Modification History

Rev.	Date	Originator	Comment
A	09/04/97	Fred Niemi Frank Pittenger	First Draft
B	09/26/97	Fred Niemi	Revised Event Model to provide better/easier filtering capability, updated per review meetings
C	12/08/97	Fred Niemi	Updates from EDS class in San Jose includes: changing Enumeration Service to Event Atom Service, Providing C++ interface, Added control events to event category, added sent time to event, added callback in CMFEventSource, changed event clearing model, defined Class Loader interface, updated title to reflect documents contents
D	03/23/98	Fred Niemi Larry Metzger Ken Chambers	Updated class names to follow BG naming conventions - includes updating package name and removing redundancy from class names, changed EventAtom interface to use Java Resource Bundles, Added definition of "query" interface to EDS updated EventSource JAVA class to add name for administration updated EventBase to allow a null resource list, combined getProbableCause and getRecommendedActions method, added event correlator ID field Updated to include EDS process startup and shutdown. Updated to include EDS tracing and logging.

Definitions

This section defines words, acronyms, and actions which may not be readily understood.

CMF	Cisco Management Framework. Components that make up the framework used by all Cisco Network Management applications.
CNMS	Cisco Network Management System
CORBA	Common Object Request Broker Architecture. Open distributed application architecture proposed by the Object Management Group (OMG). See www.omg.org for more information.
EDS	Event Distribution Service. The event distribution of the Cisco Management Framework.
Event consumer	process that is the receiver of events
Event source	process that is the generator of events.
TBD	To Be Determined

1.0 Functional Overview

1.1 System Overview

This document describes in detail the design for EDS V1.0 and NESTOR V1.0.

EDS V1.0 provides distribution from creators of events, called event sources, to receivers of those events, called event consumers. EDS V1.0 allows the event consumers to specify which events it wants to receive.

NESTOR V1.0 provides network equipment and services status reporting. NESTOR V1.0 "rolls up" related events and displays that information to the user. From this displayed information, the user can easily determine the problem and perform required operations to fix the problem. This "roll up" of events are called exceptions. NESTOR V1.0 allows the specification of what events are to considered for exceptions. It allows the creation of user profiles. These user profiles define what exceptions are to be displayed at the user's exception display console. The users can manage the exceptions by acknowledging them when they are being worked on, marking them as resolved when the problems have been fixed and removing them from the display when the problem has been verified as being fixed. NESTOR also provides for automatically resolving of exceptions by receiving "clearing" events. If all the events rolled-up into an exception are either cleared by clearing events or are clearing events, then the exception is marked as resolved.

NESTOR V1.0 allows lookup of historical exceptions and events to aid in problem determination.

1.2 Components

1.2.1 Description

EDS V1.0 Phase 1 comprises of the following components:

Event Distribution System (server)

Atom Service

EDS Manager

EDS Class Loader

EDS V1.0 Phase 2 comprises of the following components:

Event Logger

Event Logger Display

Event to Trap Converter

NESTOR V1.0 comprises of the following components:

Exception Server

Exception Display

1.2.2 Event Distribution System

The Event Distribution System (EDS) is used to distribute event messages from the creator of the events, the Event Sources (ESs), to the recipient of the events, the Event Consumers (ECs). The event is a data packet or message that contains pertinent information associated with an incident in the things being monitored/managed. Each EC registers an Event Profile (EP), or filter, with the EDS that describes the events the EC is interested in.

Event Message

The Event Message will be implemented as a JAVA class. For C++ event sources, the event message is an IDL data structure. This IDL data structure is sent to EDS. The event sources can provide a JAVA class which can read the IDL data structure and create an instance of the Event Message JAVA class. For JAVA event sources, they can create the Event Message JAVA class and send that directly. After the event JAVA class is initialized, it is serialized and sent through the EDS. It is de-serialized in the EDS and sent through the event filter. If the event passes the filter, the serialized event is sent to the interested EC. The JAVA class is de-serialized in the EC and then the event is processed in the EC. Not only is syntactical information sent with the event message, such as event creation time and event ID, but semantical information can also be sent, such as methods to display user readable message or conversion to useful traps, can also be sent.

EDS Server

The EDS server is part of Phase 1. It maintains the logical connections between the source and consumers.

Atom Service

The Atom Service is part of Phase 1. It maintains the definition of the atoms used to define the elements of the events. It keeps the definition of each atom as well as a locale correct message that can be displayed about the atom. It can be dynamically updated so that the system does not have to be stopped and started to define new atoms that appear in the event message

EDS Manager

The EDS Manager is part of Phase 1. It keeps track of all the CORBA objects that are part of the EDS system. It allows the EDS system to be administered.

EDS Class Loader

The EDS Class Loader is part of Phase 1. Since the filters provided by the event consumers and the event messages themselves are instances of JAVA classes, a class loader is needed to ensure that EDS and the event consumers can get access to the JAVA class files.

Event Logger

The Event Logger (EL) is part of Phase 2. It maintains the current event messages that flow through the EDS. This component provides search capability on these events. Searching and grouping of historical events will aid in problem determination.

Event Logger Display

The Event Logger Display (ELD) is art of Phase 2. It is used to display events from the event

logger.

Event to Trap Converter

Event to Trap Converter (ETC) is part of Phase 2. It is used to send traps to a NMS such as HP OpenView. This service receives events and then class the toTrap() method of the event message. It then constructs a trap from the returned information of that method, and then sends it to the configured NMS.

1.2.3 NESTOR

Exception Server

The Exception Server (ExSr) is used to create and process the exceptions that aid in managing the network.

Exception Display

The Exception Display (ExD) is used to display the exceptions and provide user the ability

to control the state of the exceptions

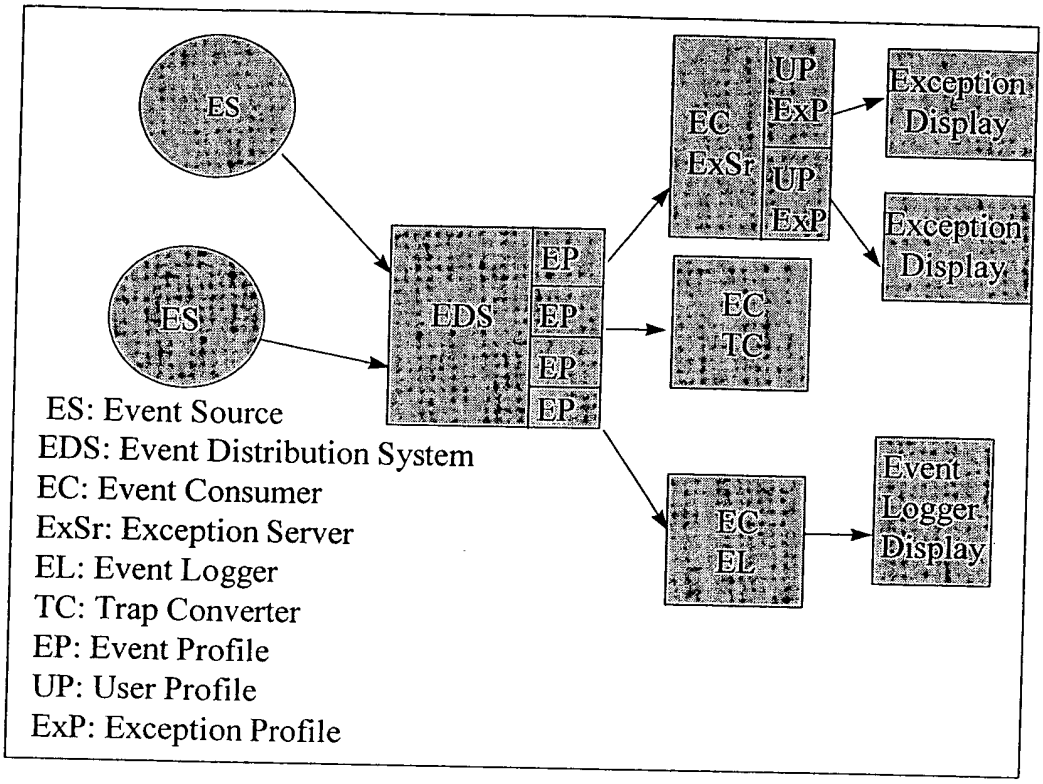
start/run tools to aid in the diagnose of the exceptions

print selected exceptions

cut and paste exceptions into email and other documents

etc.

1.3 System Block Diagram



NOTE: Event Source (ES) in above diagram is not part of EDS V1.0 or NESTOR V1.0

1.4 Features Not Addressed

Filtering on Problem Source

Current Event Display

Warning as a severity type did not make sense, therefore we are not implementing it.

1.5 Testability Considerations

TBD.

1.6 Manufacturing Considerations

TBD. Licensing.

1.7 Compliance Requirements

TBD.